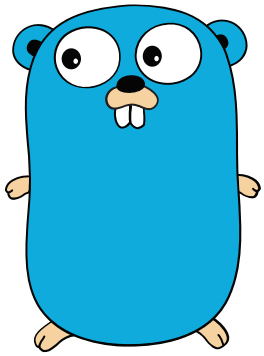
1 Get Started

Welcome to the exciting world of programming with Go. This chapter introduces the language and creates a workspace where you can run your first Go program..

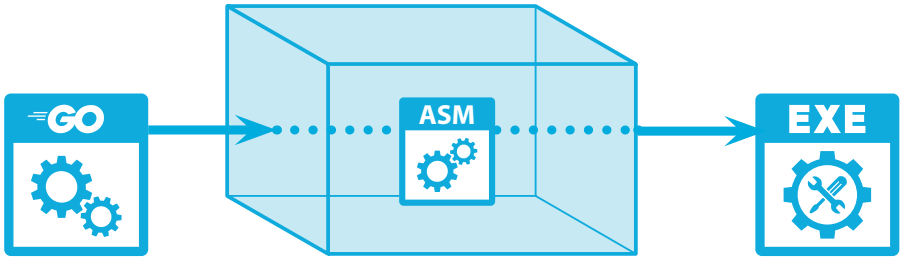
Meet the Go Language



The Go gopher – the iconic mascot of the Go programming language.

Go is a free open-source programming language created at Google by Robert Griesemer, Rob Pike, and Ken Thompson – best known for development of the Unix operating system. Google released version 1.0 of the Go language ("golang") in March 2012, since when it has gained widespread popularity.

Go programs are written in plain text, then compiled into machine code by the Go compiler to produce an executable binary version.



The aims of the Go programming language are to be expressive, fast, efficient, reliable, and simple to write. Some programming languages, such as C or C++, are fast and reliable but not simple. Conversely, other programming languages, such as Java or Python, are simple to write but not so efficient.

Go is similar to the C programming language in many ways and is sometimes referred to as a "C-like language" or "C for the 21st century". But Go is much more than that as it adopts good ideas from many other programming languages, yet avoids features that lead to complexity or unreliability.

Perhaps most importantly, Go introduces the ability to take advantage of multi-core CPU processing for concurrency using "goroutines" and "channels". This provides the possibility for the computer to deal with several things at the same time.

Although the Go language does not have the class structures found in Object Oriented Programming (OOP) languages, such as C++ or Java, its features do provide some degree of encapsulation, inheritance, and polymorphism – the three cornerstones of OOP.

With so many programming languages to choose from you may be wondering why choose to learn Go programming – so here are some of the advantages that Go offers:

Simple Syntax

The Go language is concise, like Python. It's as simple to write as Python but is more efficient, like C++. This enables you to write code that is easy to read and maintain.

Compiled Language

The Go source code is compiled to binary machine code that can be read directly by the computer, instead of being interpreted every time a program runs. This enables the Go programs you write to run faster than programs written for interpreted languages, like Python or PHP.

The Go Compiler

The Go compiler is fast and provides additional benefits, such as code optimization and error checking – it can detect unused variables in your code, missing imports that your code requires, and mistyped or invalid code. The Go compiler can also generate executable binaries for other operating systems. This enables you to compile your source code to run on multiple machines.

Concurrency

The Go language provides inherent support for concurrency with goroutines and channels. This enables you to write multi-threaded programs that could perform multiple tasks at the same time.

Garbage Collection

Automatic memory management is a key feature of the Go language. Its garbage collector runs concurrently with the program. This enables you to write program code without any concern for memory leakage.

Static Typing

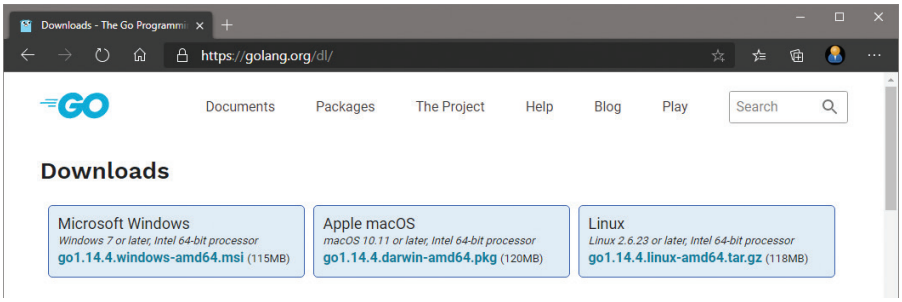
Go is a statically typed language in which variables are explicitly declared to be of a particular fixed type. This enables errors to be caught early in the development process.

You may well recognize other advantages as you gain experience with the Go language, but now it's time to get started...

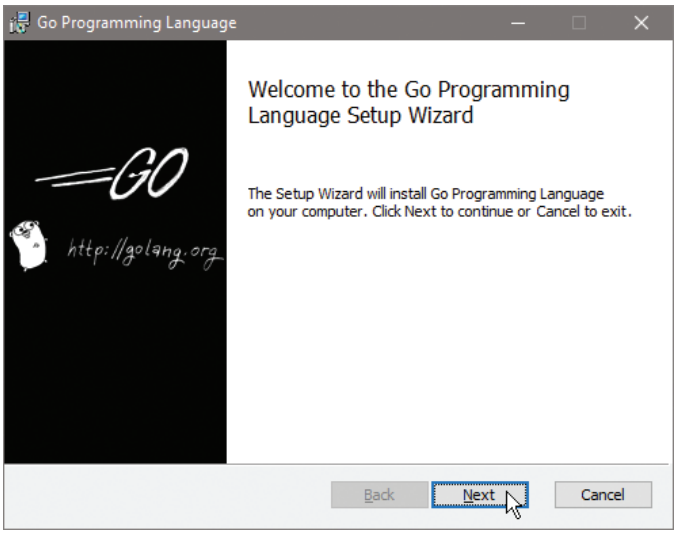
Install the Go Tools

To get started with the Go programming language you must first install the Go tools on your PC. These allow you to build, run, and test programs written in the Go language. The Go tools are supplied together with lots of standard packages of useful trusted code that you can import into your own programs. The Go language installers are available for Windows, macOS and Linux.

1 Open a web browser and visit **https://golang.org** then download the appropriate installer for your system



2 When the download has completed, run the installer to launch the "Go Programming Language Setup Wizard"

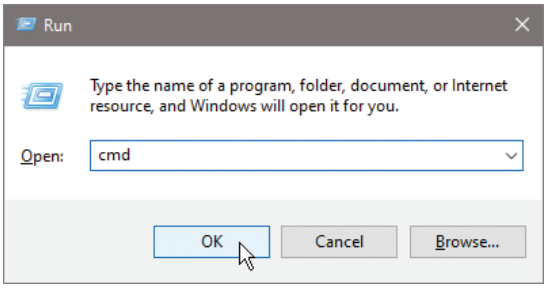


3 Click **Next** to continue, then accept the license terms

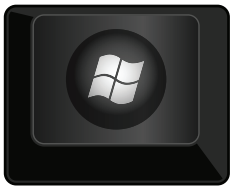
4 Accept the suggested **Destination Folder** (at **C:\Go** on Windows), then click **Install** to complete the installation

The Go installer for Windows should automatically add Go to your system path to make the Go tools available at a Command Prompt.

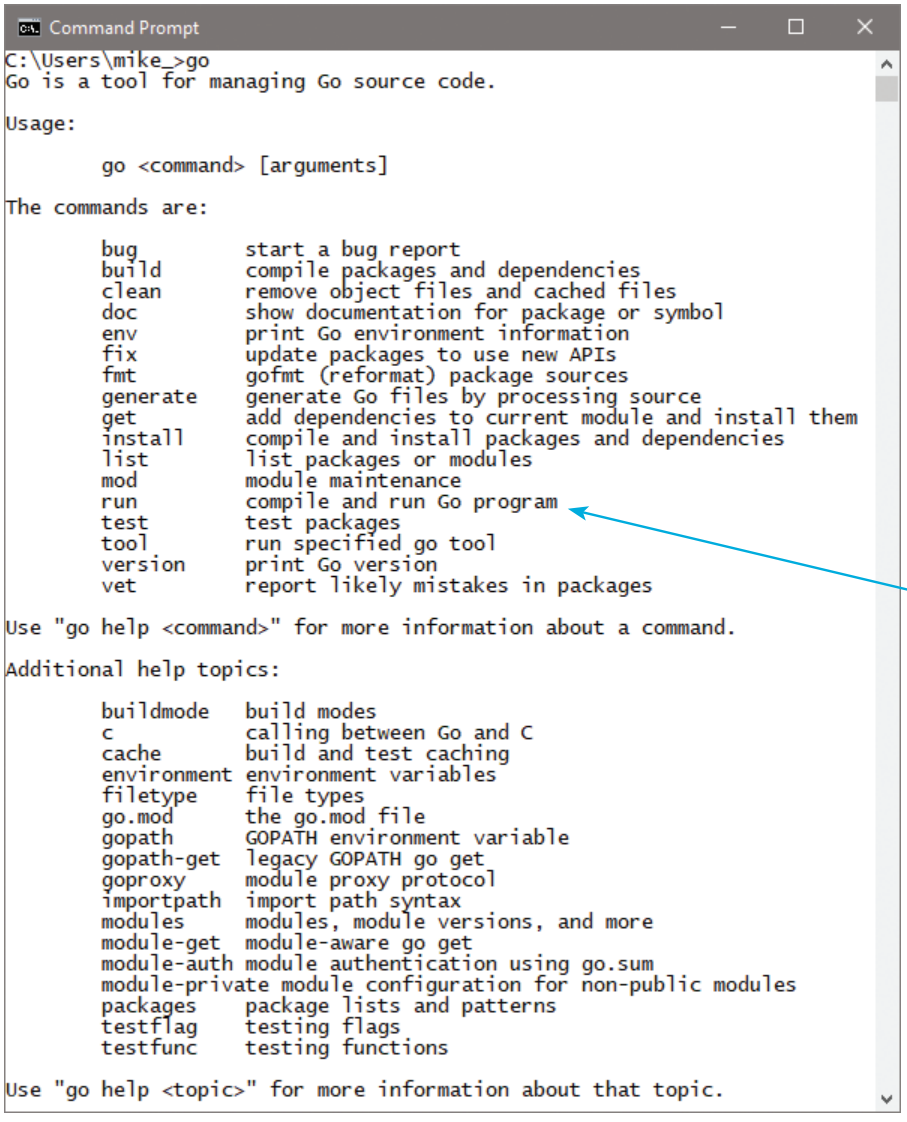
5 To test that the installation was successful first open a Terminal window – on a Windows system press **Winkey+R** together, to open a "Run" dialog, then enter **cmd** to open a "Command Prompt" window



The "Winkey" is the keyboard key labeled with the Windows logo.



6 At the command prompt, type the command **go** then hit **Enter** to see a list of Go tool commands



Although there are quite a few Go tools you will mostly use only the **run** tool to compile and run your programs.

Create the Go Workspace

When you install Go, the installer sets a number of Go environment variables. For example, the directory (folder) location of the Go tools is stored in a **GOROOT** environment variable – by default, at **C:\Go** on a Windows PC, and at **/usr/local/go** on Linux and macOS.

The installer also sets a **GOPATH** environment variable for the location of your workspace. By default this is a directory named "go" within your home directory. For example, on a Windows PC its path is **C:\Users*user_name*\go** and on Linux systems its path is **/home/*user_name*/go**, and on macOS its at **/Users/*user_name*/go** – but the installer doesn't actually create any directories.

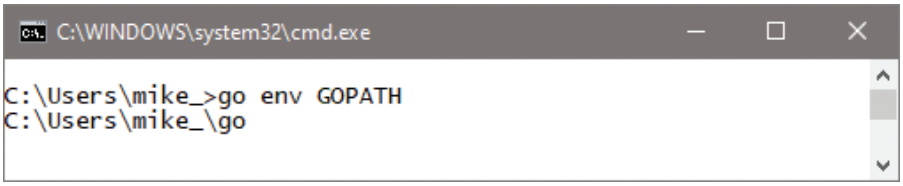
To create the workspace you can simply add a directory named "go" in your home directory. You must then add sub-directories named "bin", "pkg" and "src" within the workspace directory – all your Go programs can then be created inside the "src" directory.

It's useful to have a shortcut on your desktop that will open a command-line in this sub-directory to easily run your programs.

The **go\src** folder is where you will save the programs you write. The **go\bin** and **go\pkg** will be used later to store executable files and package archives.

1. On a Windows PC open a Command Prompt window, as described on page 11, or open a Terminal window

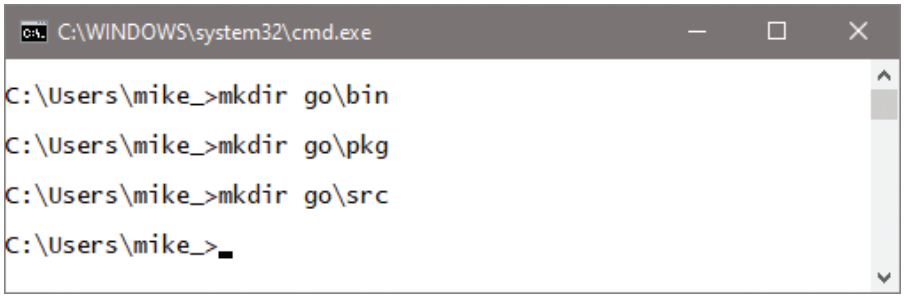
2. Enter the command **go env GOPATH** to see the current expected workspace location



3. Next issue a **mkdir** command to create the "go" workspace directory at the location specified by the **GOPATH** environment variable

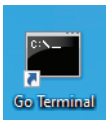
The **mkdir** command name is simply short for "make directory".

4 Now issue further **mkdir** commands to create the "bin", "pkg" and "src" sub-directories in the workspace directory

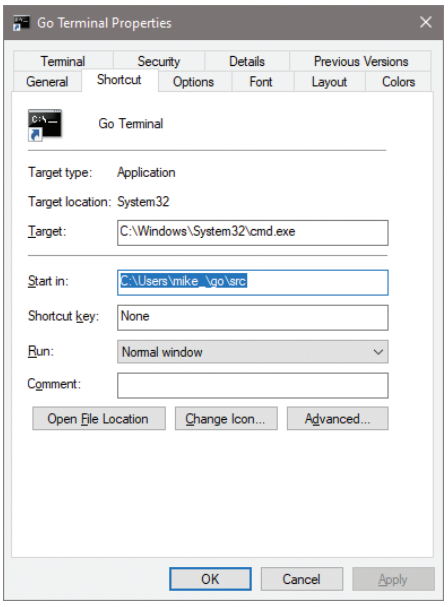


5 Right-click on a Windows desktop and select **New**, **Shortcut** to open a "Create Shortcut" dialog

6 Enter **cmd** as the location and click **Next**, then enter **Go Terminal** as the name and click **Finish** to create a shortcut icon on your desktop



7 Right-click on the shortcut icon and select the **Properties** item, to open its "Properties" dialog



8 Choose the **Shortcut** tab, then enter the location of your "src" folder into the "Start in" field

9 Click **Apply**, **OK** to close the dialog, then double-click the shortcut icon to open a command-line in your "src" folder

Go programs can be written in a plain text editor, such as Windows' Notepad app. It's useful to have a desktop shortcut that opens a text editor in your "src" directory folder. Repeat the steps on this page but in step 7 enter **notepad** as the location and **Go Editor** as the name, to create another useful desktop shortcut.

Write a Go Program

All Go programs start as plain text files that are later compiled into actual runnable programs. This means that Go programs can be written in any plain text editor, such as the Windows' Notepad app or the Nano app on Linux.

Follow these steps to create a simple Go program that will output the traditional first program greeting:

hello

1 Create a sub-directory named "hello" in your "src" folder

```
Go Terminal
C:\Users\mike_\go\src>mkdir hello
C:\Users\mike_\go\src>
```

main.go

2 Open a plain text editor, like Notepad, and type this code exactly as it is listed to begin a program

```
package main
```

3 Two lines below, insert this code exactly as it is listed

```
import "fmt"
```

4 Two further lines below, precisely add this code

```
func main( ) {

        fmt.Println("Hello World!")
}
```

5 Save the file in the "hello" folder, and name it **main.go** – the complete program should now look exactly like this:

```
main.go - Notepad
File  Edit  Format  View  Help
package main

import "fmt"

func main() {

        fmt.Println("Hello World!")
}
```

Notice the capital "P" in **fmt.Println**.

The arrangement of files within folders is important in Go. Each main file, and any related files, must be placed in a uniquely named folder to create a package – so here the package is named "hello".

The separate parts of the program code on the opposite page can be examined individually to understand each part more clearly:

The Package Declaration

```
package main
```

The package type is declared following the **package** keyword. All Go program code is contained in packages. You may declare your own type for a package that will be a shared library, but you must declare the package "main" if you want the code to be compiled into an executable program.

The package name and package type are two separate items – here the package is named **hello**, but the type is **main**.

The Import Declaration

```
import "fmt"
```

The keyword **import** is used to import one, or more, packages into this package to make their features available to this program. The package "fmt" comes from the Go standard library that is included in your Go installation. It provides the **fmt.Println( )** function that is used to output text in this program. Note that the package name must be enclosed in double quote characters. When importing multiple packages the list of package names must be enclosed within parentheses and each name must appear on its own line, like this:

```
import (
  "fmt"
  "strings"
)
```

You can find the Go standard library packages in the "src" directory of your **GOROOT** directory location, for example at **C:\Go\src**. Additionally, you can learn about each package from the official documentation at **https://golang.org/pkg** – an invaluable resource.

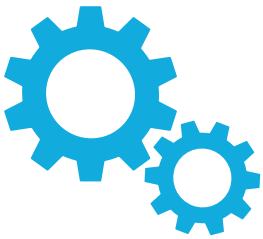
The Function Declaration

```
func main( ){

        fmt.Println("Hello World!")
}
```

The function name follows the **func** keyword. It must be followed by parentheses and an **{** opening curly bracket on the same line. The function body contains statements that are the actual instructions to perform program tasks. The function body must end with a closing } curly bracket. Function names must be unique but each Go program must have a function named "main" as this is the starting point of all Go programs.

Run a Go Program



Go program code is compiled and executed using the Go tools. During program development an error-free Go program can be compiled and run by the **go run** tool. This is useful but if you want to create an executable binary file version, that can be executed repeatedly and distributed, you can use the **go build** tool.

1. Open a Command Prompt or Terminal window in your Go "src" folder

2. Enter the command **go run hello** to run the program written on page 14 – it should output the greeting

```
Go Terminal
C:\Users\mike_\go\src>go run hello
Hello World!

C:\Users\mike_\go\src>
```

3. Next, enter the command **go build hello** to make an executable binary version of the program on page 14 – it should add an executable file in your "src" folder

4. Now, twice enter the command **hello** on Windows, or **./hello** on Linux or macOS – it should output the traditional greeting two times

```
Go Terminal
C:\Users\mike_\go\src>go build hello

C:\Users\mike_\go\src>hello
Hello World!

C:\Users\mike_\go\src>hello
Hello World!

C:\Users\mike_\go\src>
```

On Windows systems the binary file is given the file extension **.exe**, so the file built here is **hello.exe**.